Отчет по Курсовой работе
Дисциплины “Информационные системы и базы данных”

Выполнил:
Студент группы p33082
Дробыш Дмитрий Александрович, Панин Иван Михайлович

Преподаватель:
Сагайдак Алина Алексеевна

Санкт-Петербург, 2022

Этап 1.

Предметная область:

Система
CRM (Система управления взаимоотношениями с клиентами) для производителей пива Dimond & VAN.

Описание предметной области:

CRM для Dimond & VAN — это сервис, который позволит оперативно и качественно поддерживать взаимоотношения с клиентами нашей компании. Основными преимуществами нашей системы на основе базы данных будут :
- организация клиентопотока
- учет пива и его сортов, выстраивание удобного взаимодействия с клиентами
- учет сбора мнений со стороны клиентов

1) Клиент может зайти на сайт (или программу), пройти регистрацию (ФИО, номер телефона, дата рождения, способ оплаты)
2) Изучить наш ассортимент.
3) Также может просмотреть конкретное пиво, есть возможность заказать его. Потом получает подтверждение о записи после предварительной оплаты, отслеживая статус заказа.

Наша система получает данные о клиента (ФИО, номер телефона, дату рождения, способ оплаты) и хранит информацию в базе, составляет структуру спроса на товары, что повышает нашу эффективность.

CRM хранит информацию о пиве и его сортах

Сервис обрабатывает заказы предварительной оплате на сайте (интернет-платеж, не перевод).

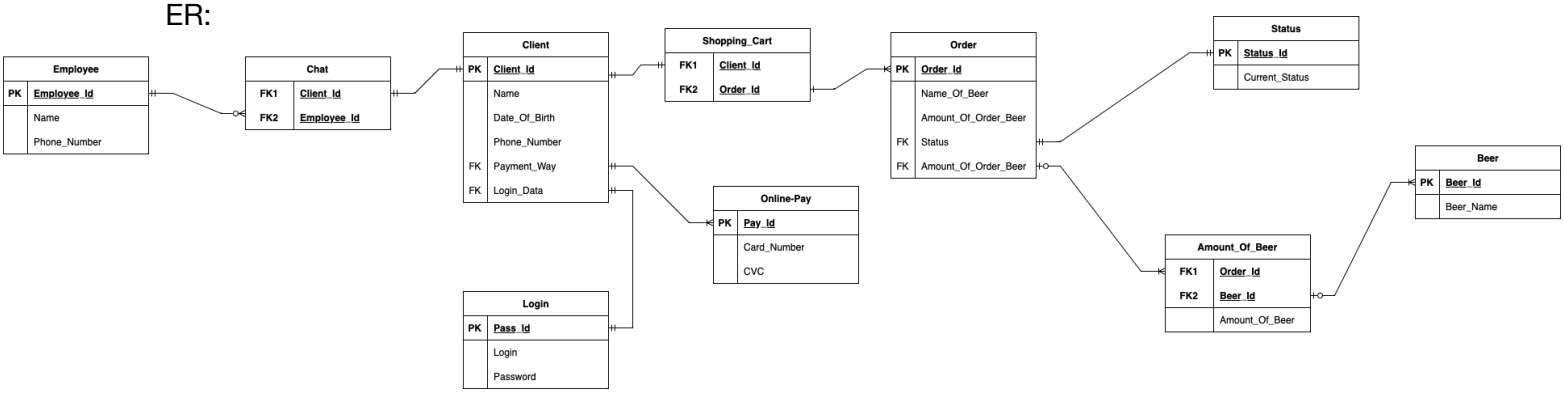
Клиенты имеют возможность связаться с нами через чат или запросить обратный звонок для получения дополнительной информации.

Бизнес-процессы.

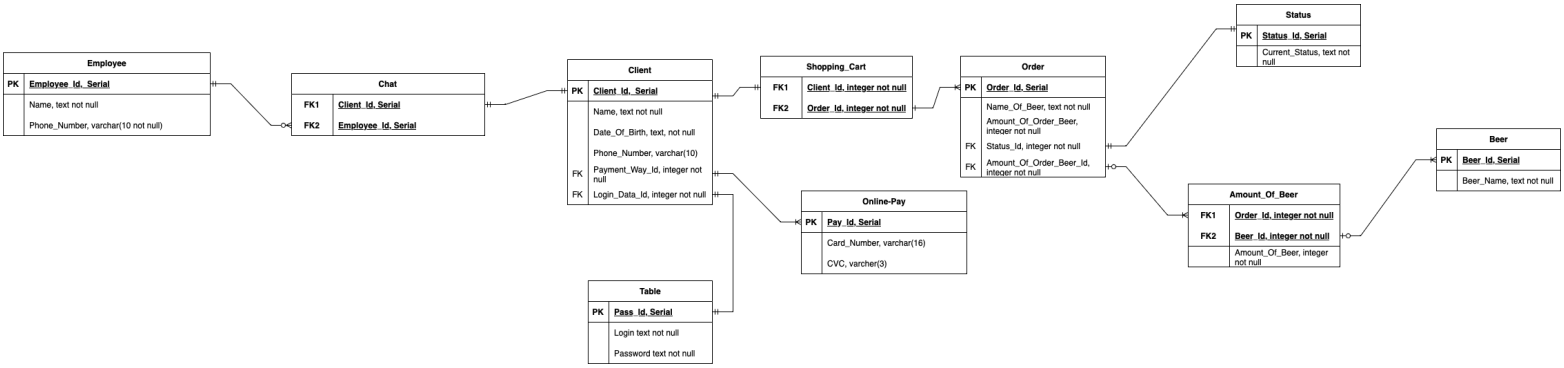
- Выбор и заказ товара:
  - Клиент получает доступ к товару нашей компании
  - Получает возможность заказать его
- Управление заказами:
  - Есть минимальное отслеживание статуса заказа
  - Присутствует информирование об изменении статуса заказа
- Учет товара:
  - Система ведет учет о наличии пива
- Оплата:
  - Если осуществляется заказ, то тогда клиент оплачивает сразу
- Коммуникация с клиентом
  - Клиент может общаться с нами через чат или заказать обратный звонок для получения дополнительной информации.
  - Наш сервис получает информацию о запросе (на чат или обратном звонке)

Этап 2:

ER:



Datalog:



Этап 3.

DDL:

```
-- create table "Employee"
create table if not exists "Employee"
(
	"Employee_id" serial primary key,
	"Name" text not null,
	"Phone_Number" varchar(10) not null
);

-- create table "Login"
create table if not exists "Login"
(
	"Pass_Id" serial primary key unique,
	"Login" text not null unique,
	"Password" text not null
);

-- create table Online_Pay
create table if not exists "Online_Pay"
(
	"Pay_Id" serial primary key unique,
	"Card_Number" varchar(16),
	"CVC" varchar(3)
```

```
);

-- create table "Status"
create table if not exists "Status"
(
	"Status_Id" serial primary key,
	"Current_Status" text not null unique
);

-- create table "Beer"
create table if not exists "Beer"
(
	"Beer_Id" serial primary key,
	"Beer_Name" text not null unique
);

-- create table "Client"
create table if not exists "Client"
(
	"Client_Id" serial primary key,
	"Name" text not null,
	"Date_Of_Birth" date not null,
	"Phone_Number" varchar(10),
	"Payment_Way_Id" integer not null,
	"Login_Data_Id" integer not null unique,
	foreign key ("Payment_Way_Id") references s333219."Online_Pay" ("Pay_Id"),
	foreign key ("Login_Data_Id") references s333219."Login" ("Pass_Id"),
	check (calculate_age("Date_Of_Birth") > 18)
);

-- create table "Chat"
create table if not exists "Chat"
(
	"Chat_Id" serial not null unique,
	"Client_Id" integer not null,
	"Employee_id" integer not null,
	foreign key("Client_Id") references s333219."Client"("Client_Id"),
	foreign key ("Employee_id") references s333219."Employee"("Employee_id")
);

-- create table "Order"
create table if not exists "Order"
(
	"Order_Id" serial primary key,
	"Status_Id" integer not null,
	foreign key ("Status_Id") references s333219."Status"("Status_Id")
);

-- create table "Amount_Of_Beer"
create table if not exists "Amount_Of_Beer"
(
	"Order_Id" integer not null,
	"Beer_Id" integer not null,
	"Amount_Of_Beer" integer not null,
	foreign key ("Order_Id") references s333219."Order" ("Order_Id"),
	foreign key ("Beer_Id") references s333219."Beer" ("Beer_Id")
);

-- create table "Shopping cart"
create table if not exists "Shopping_Cart"
```

```
(
	"Client_Id" integer not null,
	"Order_Id" integer not null,
	foreign key("Client_Id") references s333219."Client"("Client_Id"),
	foreign key("Order_Id") references s333219."Order"("Order_Id")
);
```

DML:

```
-- insert beer sorts
insert into s333219."Beer" ("Beer_Name") values
	('Pale Ale'),
	('Brown Ale'),
	('Stout'),
	('Porter'),
	('Pale Lager'),
	('Dark Lager'),
	('Lambic'),
	('Wheat beer unfiltered pale'),
	('Wheat beer unfiltered dark'),
	('Wheat beer filtered pale'),
	('Wheat beer filtered dark'),
	('Altbier'),
	('Pilsner'),
	('Kölsch');

-- insert job status
insert into s333219."Status" ("Current_Status") values
	('Accepted'),
	('In Work'),
	('Denied'),
	('Finished'),
	('Waiting');

-- insert workers
insert into s333219."Employee" ("Name", "Phone_Number") values
	('Panin Ivan', '9213452835'),
	('Drobysh Dmitry', '9319561585'),
	('Orekhov Sergey', '9665735021');

-- insert Login Data
insert into s333219."Login" ("Login", "Password") values
	('ChukhoninVan', 'LanaLanaGoslya'),
	('NorthCapDiamond', 'Boss_of_this_Data_Base'),
	('Young', 'Young');

-- insert Online_Pay
insert into s333219."Online_Pay" ("Card_Number", "CVC") values
	('1234567890987654', '123'),
	('2345678909876543', '234');


-- insert Clients
-- you should do it after others...
insert into s333219."Client" ("Name", "Date_Of_Birth", "Phone_Number", "Payment_Way_Id",
"Login_Data_Id")
	values ('Ivan Chukhonin', '16/06/2003', '9999954378', 1, 1),
```

```
        ('Drobina', '02/07/1971', '9979654378', 2, 2);
        --('Child', '01/01/2015', '9999954378', 1, 3)
```

Functions:

```
-- create fucntion for calculating the age

-- If func already exists, get rid of it
drop function if exists calculate_age;
create function calculate_age(date_of_birth date) returns integer AS $$
declare
   age integer;
begin
   age := extract(year from current_date) - extract(year from date_of_birth);

   IF (extract(month from current_date) < extract(month from date_of_birth)) or
      (extract(month from current_date) = extract(month from date_of_birth) and
       extract(DAY from current_date) < extract(DAY from date_of_birth)) then
       age := age - 1;
   end IF;

   return age;
end;
$$ LANGUAGE plpgsql;



-- test
select calculate_age('1990-05-15') as age;
```

Hard Drop:

```
drop table if exists s333219."Amount_Of_Beer" cascade;
drop table if exists s333219."Beer" cascade;
drop table if exists s333219."Chat" cascade;
drop table if exists s333219."Employee" cascade;
drop table if exists s333219."Login" cascade;
drop table if exists s333219."Client" cascade;
drop table if exists s333219."Status" cascade;
drop table if exists s333219."Shopping_Cart" cascade;
drop table if exists s333219."Online_Pay" cascade;
drop table if exists s333219."Order" cascade;
```

Triggers:

```
-- trigger for deleting users
create or replace function delete_client_trigger()
RETURNS trigger AS $$
begin
   delete from s333219."Online_Pay" where "Pay_Id" = old."Payment_Way_Id";
   delete from s333219."Login" where "Pass_Id" = old."Login_Data_Id";
   delete from s333219."Shopping_Cart" where "Client_Id" = old."Client_Id";
   delete from s333219."Chat" where "Client_Id" = old."Client_Id";
   RETURN old;
end;
$$ language plpgsql;
```

```
create trigger delete_client_trigger
before delete on s333219."Client"
for each row
execute function delete_client_trigger();


-- replace employee in chat
create or replace function replace_employee()
returns trigger as $$
begin
   update s333219."Chat" set "Employee_id" = (select "Employee_id" from s333219."Employee"
where "Employee_id" <> OLD."Employee_id" limit 1) where "Employee_id" = OLD."Employee_id";
   return new;
end;
$$ language plpgsql;

create trigger replace_employee_trigger
before delete on s333219."Employee"
for each row
execute function replace_employee();


-- remove shopping cart
create or replace function delete_shopping_cart_trigger()
RETURNS trigger AS $$
begin
   delete from s333219."Order" where "Order_Id" = old."Order_Id";
   RETURN old;
end;
$$ language plpgsql;

create trigger delete_shopping_cart_trigger
before delete on s333219."Shopping_Cart"
for each row
execute function delete_shopping_cart_trigger();
```